Automation of Distributing the Teaching Load to Increase the Effectiveness of Planning of Work of the Teaching Staff

https://doi.org/10.3991/ijep.v10i2.11614

Leonid L. Khoroshko (✉), Maxim A. Vikulin
Moscow Aviation Institute (National Research University), Moscow, Russian Federation
khoroshko@mati.ru

Abstract—The article describes the developed algorithm for the formation of the training load, covering the whole cycle of operations, including: obtaining the initial data and its processing, aggregating the disciplines, calculating the hours, forming a list of teachers, and downloading the information as the form of a spreadsheet.

Keywords—Training load, the organization of the learning process, algorithms of automation

1 Introduction

The scheduling of student classes is one of the most difficult and important tasks in organization and management of the educational process in higher educational institutions. A huge amount of work needs to be done for correct timetable scheduling:

- Curriculums review in order to identify the disciplines to be studied in the current semester, as well as to determine the number of hours required for their study
- Distribution of classes by type of in-class work [1]
- Review of the academic group lists and dividing the classes into subgroups or combining them into rounds, in accordance with established standards
- Review of the student recruitment plan (in case an autumn semester schedule is being drawn up) and distributing them by cohorts or subgroup for each class
- Assignment of the teacher to each class exercise
- Distribution of classes by buildings and classrooms, taking into account their equipment and requirements for classrooms [2]
- Drawing up a general schedule according to the collected and processed data, taking into account the wishes of teachers.

With such a large amount of work, the probability of error is also increased, especially if all work is done manually. At the same time, it is essential to recall that an error in scheduling can lead to disruption of classes, which adversely affects the educational process. Moreover, such work is carried out every semester, which further emphasizes the relevance of this problem.

The formation of the training load is an integral part of preparation to forming a schedule, as the formed load represents disciplines split into classroom sessions, distributed among cohorts or subgroups, and it also establishes the connection of each session with the teacher. Thus, we can say that a properly formed load is the key to a properly created schedule [3].

Another important application of load is calculation of the hours that a particular teacher spends on teaching, that is actual individual load of the teacher. This number is necessary to calculate the rate of the teacher, and therefore to determine his or her salary.

Due to the specified priority of this issue, it was decided to automate a part of the work that will allow to skip some steps of preparation to forming a schedule and simplify others.

2 Algorithm for the Formation of the Load

Algorithm for the formation of the load consists of several parts, including data collection, its preparation, aggregation, processing and output to the resulting form. Thus, the following steps can be distinguished in the algorithm:

- Obtaining data of the form and its processing
- Obtaining and forming auxiliary lists
- Obtaining a list of academic groups
- Obtaining a general list of students;
- Obtaining a list of disciplines depending on the calculation method
- Forming a consolidated list of disciplines indicating academic groups
- Aggregating disciplines by key parameters
- Forming cohorts and subgroups for classroom sessions
- Calculating the hours allocated to other forms
- Forming a list of the teaching staff
- Downloading the received information to the template.

After submitting the form, the identifier of the department engaged in educational activities and the number of the semester that starts the academic year are transferred to the function of calculating the load. In this respect, the semester number starts from the current school year. This means that the first value of the school year in the list will always transfer one to the function, and the second and third items will transfer three and five respectively.

To form a complete list of academic groups, it is necessary to perform several actions. First you need to get all the objects of academic groups that are currently active. Then check whether they will be trained during the period for which the calculation of the load is performed. Also, it is necessary to add groups previously formed on the basis of the admission quotas [4] to the obtained list.

To get a list of academic groups that can study the disciplines that appear in the load, you must first select all the curricula with disciplines read by the departments obtained at the first step of the algorithm. To do this, the value of the plan_id field is selected from the iis_subject_source table with the exception of duplicates for all disciplines for which the education_id field is in the list of educational identifiers of departments for which the load is formed.

Then you need to get a list of identifiers of academic groups. This is done by linking the curriculum with the group in the student object. Thus, the selection of the group_id field is done with the exception of duplicates from the iis_students table, where there is the plan identifier in the list obtained at the previous step.

The resulting list of academic groups is sorted in a cycle and for each group the semester of study is increased by the value of the $semester variable. Thus, the group is transferred to the semester of training it will have during the period for which the load is calculated. The group commentary contains complete information about the training periods, therefore if the obtained semester value does not appear in this field, the group will not be taken into account when forming the load.

The next step is to form data structures similar to the objects of academic groups to store information about applicants, the number of which is determined on the basis of the admission quotas [5]. As the applicants must participate in the calculation and information about them is not yet in the database (except for the total number determined by the admission quotas), obtaining a general list of students is also divided into two stages.

First, all academic groups that are already studying at the university are processed. The identifiers of all such groups are collected in the list and the functions of obtaining students from the database are transferred. This function returns an array of all student objects that are in the list of academic groups, with the identifiers of the curricula, according to which students are trained. Then all the obtained students are sorted in a cycle, and for each object a semester field is created, to which the corresponding value of the semester of study from the academic group is added.

Then all the objects of the $groups array containing the prefix “r” in the key are sorted. For each of these elements, student objects are created in the number equal to the count field and containing the group key, plan identifier and semester of study. Each created student object is added to a common $ students array.

All received disciplines are sorted in a cycle in which an array of students studying this discipline is formed for each element. The formation occurs due to the connection of the student object with the curriculum and the semester of study: if the discipline and the student have the same plan identifier, and the discipline study semester is equal to the current or subsequent semester of a student, then such a student will study this discipline within the load calculation period.

After forming the lists of students, the array of disciplines is checked for empty elements. If they are found, they are deleted. After that, the array of disciplines is sorted by key, and based on the list of identifiers of the obtained disciplines, two lists are formed: $ subjects Data and $ subjects Data List.

The summary list is built on the basis of the $ subjects Data List array, which contains objects of disciplines sorted by name. For each element in the cycle there are groups that study this discipline, for output to a file.

The search of the required academic groups is performed by sorting all the elements of the $ subjects array. In this case, a list of identifiers is formed, that is necessary for displaying information about the groups.

The resulting list is sorted in the cycle where an array containing information about the discipline is created for each group identifier: academic unit, name, semester of study, place of sessions, number of credits, hours of lectures, laboratory and practical sessions, hours of control of independent work, information about a course paper or a course project, form of control, a reading department, course identifier in the e-learning system and curriculum code.

For the work of the function of aggregating disciplines by key parameters, the $ subjects Data array is used, it is obtained at the fifth step of the algorithm and contains a list of disciplines sorted by the semester of study.

First, an empty $aggregate array is created, that is used to store the unique key according to which aggregation will be performed. Then, all disciplines are sorted in the cycle, for each of disciplines there is formed a key consisting of a training unit, name, course identifier in the e-learning system, number of credits, a semester of study, a reading department, place of sessions, information about a course paper or a course project, hours of lectures and practical classes, laboratory work and independent work control, forms of control and forms of training. Such a multitude of fields allows us to identify the criterion according to which the aggregation should take place.

The key is compiled by concatenating the values of all named fields with a specific separator. In this respect, the separator should be chosen in such a way so that there is no likelihood that it will occur inside any data (for example, in the name of the discipline). As a result, a construction consisting of concatenation of two characters was chosen as a separator: less and more characters ("<>").

After the key is generated, there is made a check for its existence in the $aggregate array. If the result is negative, an element is created for this key, which is an empty array. Then the discipline identifier is added to this element. Thus, after the completion of this cycle, there is an array, the keys of which are aggregation criteria, and the elements are lists of identifiers of disciplines that fall under this criterion.

Then, based on the result, another array is formed, it will have the same keys, but the elements will be lists of students studying disciplines that satisfy the aggregation criteria. For this, for each criterion all discipline identifiers are sorted, according to this there are obtained elements of the $subjects array formed at the fifth step.

The resulting $ aggregate Subjects array is used in the implementation of the eighth and ninth steps of the algorithm. For this, the elements are sorted, during this process the indicated steps are performed for each aggregation criterion, and the preparatory stage is done. Within this stage, first of all, the criterion is split according to a given separator into an array of elements for their further use. Then, all students studying disciplines are sorted in order to determine the list of identifiers of academic groups where these students study. After that, the $ study Groups array is formed, it contains group objects taken from $groups and with identifiers that are contained in the received list.

At the following step, the first four types of classroom sessions are sorted: lecture, practical and laboratory sessions, as well as control of independent work. First of all, an object of a discipline is checked for the presence of hours. Types of sessions that do not have hours are skipped, since such sessions are not held, and therefore do not go to the load.

Then the calculation formula is determined; for the indicated types of sessions it represents the maximum number of students who may be present at this session. First, the formula is the default one for this type of work. After that, additional formulas obtained at the second step are sorted, during this process each element is checked whether it can redefine the current formula [6].

Thus, after all formulas related to this type of sessions are sorted, the $formula variable will have the value of the last element for which the described condition was fulfilled. Such actions will provide the correct result due to the fact that the most specific formula should be applied, as well as due to a certain sorting of the formulas, that arranges the records received from the database in increasing order of field specification.

For each academic group, the number of students is compared with the obtained formula, which determines whether there will be formed a cohort or whether there will be a division into subgroups.

The second operation is performed if the number of students in the group is greater than the maximum allowed number of students at the session, that is contained in the $formula variable. To be divided into subgroups, first we should obtain their number: the number of students is divided by the value of the formula and rounded up. Then a cycle starts, in which for each number of a subgroup decreasing from the total number to one inclusively, the number of students in this subgroup is received: the number of students not distributed among subgroups (at the first step – the total number of students in the group) is divided by the number of the subgroup and rounded up. Then the variable $ group Count (the number of not distributed students) is reduced by the result. Information about the received subgroup is recorded in the element of the $types array, that corresponds to the current type of studies, where the following data is indicated: study period (autumn or spring and the calendar year), a reading department, training unit, the discipline name, place for studies, information about the subgroup in the form of the group name and subgroup number and the number of students in it, the number of academic hours, as well as separately the number of students of the subgroup, that is necessary for calculations.

In that case, if the number of students in the academic group is less than stated in the formula, then an attempt will be made to form a cohort of several groups [7]. To do so, there starts a cycle with a precondition, that will be executed until the total number of students in the cohort is less or equal to the value of the formula. In each iteration, an output line is formed for the current group and added to the array, and the number of students is added to the total. Then it checked whether there is a subsequent item in the group list. If it is defined, then the number of students is added to the total and compared with the $formula variable. If it does not exceed the value of the formula, then the counter of the external cycle that goes through all the groups studying this discipline increases. If the total number exceeds the permissible limits, then the cycle is left with a precondition and information about the groups put together in the cohort is added to the $types array. Thus, due to the increase in the counter of the external cycle, the groups formed into the cohorts will be skipped during further processing and that will help to avoid data duplication.

At the following step information is generated about the remaining types of sessions, the hours of which are estimated. Depending on the type of a session, it is checked whether it is necessary to include it into the load. If the key field of this type of work is empty or, for practice and final certification, the training unit is not in the list of acceptable ones, the discipline will be skipped.

Then all academic groups studying this discipline are sorted, where the initial formula is calculated for each group. Before calculating the formula, the following data is substituted: the number of students, groups and credits. The number of students and credits varies depending on the group and discipline respectively, while the number of groups is always equal to one, since the calculation is done without aggregating objects (that is, separately for each group).

After calculation, the corresponding information is added to the $types array. For consultations, if there are lecture hours, the separate load is also added that equals a percentage of the lecture hours, depending on the form of training. For the final certification, in addition to the manual, there is also a load calculated by a separate certification formula, for which data is also substituted and calculated, after that the information is added to the “FC” section.

After completion of the algorithm for the formation of the load, its result is recorded in the developed Excel workbook template that is downloaded from the server to the user's computer and where the individual load is distributed directly among the teachers.

To work with the template, the Open TBS template engine is used and the template is loaded. Within the framework of this template engine, all fields indicated in the template must be defined, therefore all elements of the $types array are checked, and if an empty element is found during this process, it will be filled with an array containing the necessary associative keys and empty lines as elements.

As a result, at this stage, there is already a calculation of the load itself and a downloaded file with it; the end user can work with this file and distribute the individual load among the department staff. But for the correct determination of the number of hours allocated for a certain type of work, it is also necessary to take into account the time standards approved by the relevant order.

The resulting software can be integrated into any Learning Management System (LMS) that has administrative features for the management of the educational process, as well as into any other administrative system that has a web interface and a database filled with the necessary initial data.

3 References

[1] L.L. Khoroshko, P.A. Ukhov, P.P. Keyno. Development of massive open online courses based on 3d computer graphics and multimedia // International Journal of Engineering Pedagogy. 2019. No 9(4) pages 4-15. https://doi.org/10.3991/ijep.v9i4.10193

[2] L.L. Khoroshko, P.A. Ukhov, A.L. Khoroshko. The use CAD/CAE systems to create E-learning courses on technical subjects at university // International Journal of Engineering Pedagogy. 2018. No 8(2) pages 64-71. https://doi.org/10.3991/ijep.v8i2.8134

[3] E.S. Tolstykh, A.A. Tolstykh Automation of forming the schedule in the system of educat Article submitted 2019-09-02. Resubmitted 2019-11-29. Final acceptance 2019-11-30. Final version published as submitted by the authors. Article submitted 2019-09-02. Resubmitted 2019-11-29. Final acceptance 2019-11-30. Final version published as submitted by the authors.ional process management // The territory of science. 2014. No. 1 pages 41-48

[4] N.V. Kalyuzhny. The analysis of the process of the teaching load and academic teaching staff at the departments // Science Time. 2015. No. 6 (18) pages 199-202

[5] M.A. Vikulin, L.L. Khoroshko. Development of the service for storing the admission quotas using the MAI e-learning system // XLIII International conference for young researchers: Abstracts: M. Moscow aviation institute (national research university), 2017. 1478 p.

[6] A.V. Kukin. Mathematical model of the distribution of the teaching load in the university // Mathematical structures and modeling. 2003. No. 2 (12) pages 165-170

[7] N.V. Pyankova, I.M. Glotina, K.V. Naugolnykh. Prospects for solving the issues of automation of distribution and control of the teaching load execution at the department // Perm agrarian bulletin. 2013. No. 2 (2) pages 53-55

4 Authors

Leonid L. Khoroshko, Associate Professor, PhD, Head of the Department of System Modeling and Computer-Aided Design, Moscow Aviation Institute (National Research University). He is the author and developer of electronic training courses on technical subjects at the university. He often works as an author and reviewer of publications of international conferences (ICL, EDUCON, EDUNINE, World CIST, TALE, SDF). Email: khoroshko@mati.ru

Maxim A. Vikulin is an Assistant of the Department of System Modeling and Computer-Aided Design, senior developer of distance learning systems, Moscow Aviation Institute (National Research University).

Article submitted 2019-09-02. Resubmitted 2019-11-29. Final acceptance 2019-11-30. Final version published as submitted by the authors.